APPLICATION FOR UNITED STATES LETTERS PATENT

METHODS AND APPARATUS FOR ADDING A UNIQUE SIGNATURE TO NETWORK DATA TRAFFIC

Inventors:

Leslie V. Niles
264 Ventura Ave.
Palo Alto, CA 94306

Assignee:

Narus, Inc.
3950 Fabian Way
Palo Alto, CA 94303
A California Corporation

Entity: Small

Beyer, Weaver & Thomas LLP
P.O. Box 778
Berkeley, CA 94704
Tel: (650) 961–8300

Attorney docket no. 5465/NARSP004

METHODS AND APPARATUS FOR ADDING A UNIQUE SIGNATURE TO NETWORK DATA TRAFFIC

FIELD OF THE INVENTION

The present invention relates generally to methods and apparatuses for adding a unique signature to monitored traffic flowing across a network. This signature can be further used in the analysis and handling of the network traffic reports, given that each report can be uniquely identified.

BACKGROUND OF THE INVENTION

Computer networks provide an efficient means for transporting data between workstations or terminals on (or connected to) the network. Such networks can consist of Local Area Networks (LANs), which are generally restricted to one geographical area or location. Such networks can also include Wide Area Networks (WANs) which connect a number of machines over a larger geographic area. The Internet is an example of a widely distributed worldwide network. The Internet is a system of computer networks -- or a network of networks -- wherein users at any one computer can get information from any other computer, provided that access has been granted. The Internet was conceived by the Advanced Research Projects Agency (ARPA) of the U.S. government in 1969 and was first known as the ARPANet. The original aim was to create a network that would allow users of a research computer at one university to be able to "talk to" research computers at other universities. A side benefit of the ARPANet design provided that messages can be routed or rerouted in more than one direction. The network can continue to function even if parts of it are destroyed in the event of a military attack or other disaster (including simple down-time of component parts). Today, the Internet is a public, cooperative, and self-sustaining facility accessible to hundreds of millions of people worldwide. The Internet is thereby providing a low-cost medium for the exchange of information between certain authorized parties, across a variety of paths that comprise the overall network.

Networks commonly exchange data through a packet-based protocol such as TCP/IP. TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. TCP/IP can also be used as a communications protocol in private networks called intranets and in extranets. Each computer on a network has a copy

of the TCP/IP program. TCP/IP is a two-layered program. The higher layer, Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, or Internet Protocol (IP), handles the address part of each packet so that it arrives at the proper destination. Each gateway computer on a network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, the packets will be reassembled at the destination to form the complete message.

TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as the sending a Web page) by another computer (a server) in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point (or host computer) in the network to another point or host computer. TCP/IP and the higher-level applications that use TCP/IP are sometimes said to be "stateless" because each client request is considered a new request that is unrelated to any previous request (unlike ordinary phone conversations that require a dedicated connection for the call duration). Note that the TCP layer itself is not stateless as far as any one message is concerned. The TCP connection generally remains in place until all packets in a message have been received.

Another IP protocol includes UDP (User Datagram Protocol). UDP is a communications protocol that offers a more limited amount of service for messages exchanged between computers in a network. UDP is an alternative to the Transmission Control Protocol (TCP). Like TCP, UDP uses the Internet Protocol to actually transmit a data unit (called a datagram) from one computer to another. Unlike TCP, however, UDP does not provide for dividing a message into packets (datagrams) and reassembling the packets at the other end. More specifically, UDP does not provide for sequencing of the packets that carry the message data. As a result, an application program that uses UDP must be able to confirm that the entire message has arrived and that the packets are in the proper order. Network applications that need to save processing time might prefer UDP over TCP. This would include networks that have very small data units to exchange, and therefore will have a relatively small amount of message reassembling to perform. Most network traffic utilizes IP, with roughly 80% using TCP, and 20% using UDP. Still other non-IP methods or

protocols might also be used to transfer data over a network. Such non-IP methods are usually proprietary to various companies and/or developers.

As the through-put rate of distributed computer networks increases, data packet (and other types) of transmissions are being used to send content that might be sold or subscribed to a user. The content might include (for example) videos, movies, television shows, concerts, and/or phone conversations. The content might be priced on an item-by-item basis, or the overall value of the content might depend on the amount of data sent. Note that older phone systems (and the like) were set up from their onset to accommodate billing concerns. For instance, a connection was made from a known origination point to a known destination point through a well-defined switching network. The connection was monitored and the usage (time or otherwise) was billed accordingly based upon switching logs which were later analyzed.

For Internet exchanges, one type of interchange that might be monitored for billing purposes would be sessions over the "session layer" (sometimes called the "port layer") of the Open Systems Interconnection (OSI) communications model. The session layer manages the setting up and taking down of the association between two communicating end points that is called a connection. A connection is maintained while the two end points are communicating back and forth in a conversation or session of some duration. Some connections and sessions last only long enough to send a message in one direction. However, other sessions may last longer, usually with one or both of the communicating parties able to terminate the session. For Internet applications, each session is related to a particular port. This port is a number that is associated with a particular upper layer application. For example, the HTTP (Hypertext Transfer Protocol) program, or daemon, commonly uses port number 80. A daemon is a program that runs continuously and exists for the purpose of handling periodic service requests that a computer system expects to receive. The daemon program forwards the requests to other programs (or processes) as appropriate. The port numbers associated with the main Internet applications are referred to as "well-known" port numbers. Most port numbers, however, are available for dynamic assignment to other applications.

The Internet or any other IP network, however, does not lend itself well to the billing needs associated with monitoring data packets, sessions, or the like. Problems exist with

simply superimposing, or layering, a billing structure on top of such IP networks. As already mentioned, any given message is generally broken down into data packets that can be transmitted over any number of routes or paths that comprise the network. In order to track such data packets, at least one network path needs to be monitored. When monitoring computer network traffic, it is usually necessary to monitor at several different points in the network, since there is no single point through which all the traffic passes. Since the routing of network traffic is unpredictable and changes in real time without any centralized coordination, these multiple monitoring points will often see some of the same traffic. Note that reports are generated pertaining to the monitored traffic. The same traffic passing different monitoring points will result in duplicate reports being generated for that traffic. In certain cases, it might be possible to select monitoring points such that different monitors might never see the same traffic. However, it is often inconvenient and/or inefficient to monitor at such points.

When analyzing the reports collected by the various monitors, the duplicates resulting from monitoring the network at multiple locations will lead to over-reporting some of the traffic. This is a problem for any kind of traffic-based billing system. Moreover, such over-reporting can make statistical traffic analysis of applications unreliable. Because of the dynamic nature of network routing, there are presently no simple or effective ways to identify (and eliminate) likely occurrences of duplicate data. In order to delineate different packets, one solution might including transmitting enough of the contents of each network packet to a central collector in order to be able to identify any duplicates. However, this becomes impossible or impractical because the solution requires too much information to be transmitted from each of the monitors to a central location. In general, collecting and analyzing the various reports at a centralized location requires reducing, as much as possible, the amount of information contained in those data reports.

Still another solution might include having the various monitors communicate with each other in order to coordinate the processing (and elimination) of duplicate traffic. This is impractical, however, as any such implementation would not scale for use with large networks. For instance, if there were N different monitoring locations, then each monitor would need to communicate with N-1 peers in order to effect this coordination. The communication network between the monitors would start to rival the size and complexity of

the overall network being monitored. The amount of traffic being passed between the monitors would also be cumbersome for larger networks.

Accordingly, a solution is needed which will provide an effective way to uniquely identify the data packets flowing across a network connection. The unique identification should be derived only from what is contained within the data packet so that data monitors at different points along the network can make the same unique identification, without the need for communication between the various other monitors. The identification data might thereafter be applied, in association with specialized solution methods, to eliminate duplicate data reports from a final compiled list of reports.

SUMMARY OF THE INVENTION

To achieve the foregoing, and in accordance with the purpose of the present invention, certain methods and apparatuses are described for eliminating duplicate data reports (or sets of data reports) from a set of monitored data. This application describes methods (and apparatuses) for attaching a unique identifier to the monitored data reports

A plurality of data monitors are placed along the connection paths of a network hierarchy. The data monitors are hardware and/or software devices that are configured to communicate with a central processing device (or central collector) which evaluates and analyzes the incoming data. Each data (or network) packet contains information that can be used to generate a signature that uniquely identifies that data packet. Reports pertaining to the data packets are sent to the central collector. The reports and corresponding signatures can be used for many purposes (i.e., analysis and the like), any of which benefit from the unique identification of the network data. In particular, the signatures can be used according to certain methods to eliminate duplicate reports from a resulting list of data reports.

Each report from the monitor can include a tag, or signature, that uniquely identifies the network traffic that produced that report. The signature is determined entirely from what is observed on the network, and not from any internal state of the monitoring device (such as a local counter or clock). According to this arrangement, two different monitors observing the same network traffic can compute the same signature without coordination between them. The central collector can thereafter perform any of a variety of actions upon the signature

data. In the simplest case, the central collector can compare the signatures of the received reports and discard any that are duplicates.

The signature is formed to be as small as possible in order to facilitate its efficient transmission and storage. A minimum size of the signature is determined by the need to prevent collisions, i.e. cases where the same signature results from traffic that is actually unique, and not a duplicate. In the present application, collisions will lead to false detection of duplicates, and therefore the discarding of reports that should have been retained.

A signature meeting the above requirements is computed from a cryptographically strong hash function operating on selected parts of the network traffic that contribute to the report to which the signature is attached. A hash function takes input bits and produces output bits that vary according to the changing input bits. A cryptographically strong hash function (i.e. MD5 or the like, as discussed below) has the property that changing any bit (or part) of the input data changes the resulting output. Moreover, any random selection of input data values will, with very high probability, produce a uniform distribution of different result values. The hash function therefore allows for the reduction of a large number of bytes of data -- i.e. the portion of the observed traffic sufficient to establish the uniqueness of the observation -- to a relatively compact signature value.

For any given protocol, certain unique information within the data can be used as the input to the hash function in order to generate the unique signature. For TCP exchanges, the client address, server address, client port, server port, and initial client sequence number (and optionally the initial server sequence number) are extracted from the header information of transmitted data packets. For UDP exchanges, no sequence number information is available. This information is compiled in a standardized manner and used as the input to the hash function. The hash function generates an output value which is often much longer than necessary (for practical purposes). Accordingly, the output value can be truncated to fewer bits in order form a smaller signature and thereby speed transmission of such data. Additionally, bits pertaining to an application-level flag can be added to shift the frequency of collisions toward reports from protocols (applications) that are more frequent and away from those that are less frequent. The final collection of bits comprise the unique signature which is attached to each report (or set of reports) associated with the data. The unique

signature (or signatory) data can be used by the central collector to analyze the network, and/or to eliminate duplicate reports via comparison of the signatory data.

According to one aspect of the present invention, a method is provided for unique identification of monitored network data instances flowing across various connections between networked devices, the unique identification being derived from information contained entirely within each instance of the network data, the method comprising: using at least one monitoring device to monitor a network data instance flowing across at least one data connection; deriving from the data instance certain information which collectively provides a unique identification of the network data instance; assembling the derived information into an input string for a hash function; and using the output string of the hash function as a signature which represents a unique identifier of the network data instance.

According to another aspect of the present invention, an apparatus is provided for unique identification of monitored network data instances flowing across various connections between networked devices, the unique identification being derived from information contained entirely within each instance of the network data, the apparatus comprising: at least one monitoring device positioned to monitor a network data instance flowing across at least one data connection; a hash function device having an input string and an output string, the input string assembled from certain information derived from the network data instance, the information collectively providing a unique identification of the network data instance; wherein the output string is used as a signature which represents a unique identifier of the network data instance.

According to another aspect of the present invention, a method is provided for unique signature of monitored network data packets flowing across various connections between networked devices, the unique signature being derived from information contained entirely within each instance of the network data packet, the method comprising: using at least one monitoring device to monitor a network data packet flowing across at least one data connection; deriving from the data packet a source and destination address for the data; deriving from the data packet a source and destination port associated with the networked devices; deriving from the data packet at least one sequence number associated with data instance; assembling the derived addresses, ports, and at least one sequence number

information into an input string for a hash function; and using the output string of the hash function as the signature which represents a unique identifier of the network data packet.

These and other advantages of the present invention will become apparent upon reading the following detailed descriptions and studying the various figures and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

Figure 1 is a representative block diagram, according to one aspect of the present invention, of a hierarchical network structure having monitors for sampling network traffic and a central collector in communication with the monitors.

Figure 2 is a representative block diagram, according to one aspect of the present invention, of a portion of a network structure which shows different monitors receiving different data packets pertaining to one session.

Figure 3 is a representative block diagram, according to one aspect of the present invention, showing a client and server in communication with each other, and certain information used to establish the connection.

Figure 4 is a representative block diagram, according to one aspect of the present invention, of certain information elements being used by a hash function to create a unique signature associated with the data packet.

Figure 4A is a representative block diagram, according to one aspect of the present invention, which shows truncation and adding of other bits to the resultant signature.

DETAILED DESCRIPTION OF THE INVENTION

An invention is described herein for providing methods and apparatuses for uniquely identifying data reports by using data contained within each report. The identification data is used to generate a more compact signature which is thereby associated with the data reports.

The descriptions below include references to data monitors connected across network paths between machines (clients, servers, or the like) on a network. While not limited to such, these data monitors include hardware and/or software devices that can be unobtrusively connected across the network path, without hindering the transfer of data across the path. The data monitors can also include devices that monitor indirectly. Indirect monitoring might include reading log files or other information recorded by hardware or software devices that actually observe the network traffic. One example protocol includes SNMP (Simple Network Management Protocol), which is the protocol governing network management and the monitoring of network devices and their functions (and which is not necessarily limited to TCP/IP networks). Still another example might include Cisco Netflow Protocol.

The data monitors communicate with a central processing device (or central collector) which evaluates and analyzes the incoming data. Each data (or network) packet contains information that can be used to generate a signature that uniquely identifies that data packet. Reports pertaining to the data packets are sent to the central collector. The reports and corresponding signatures are used to assist in analysis of the network data. The signatures might also be used to eliminate duplicate reports from the resulting list of data reports.

Each report from the monitor can include a tag, or signature, that uniquely identifies the network traffic that produced that report. In a general sense, it is important that this signature be determined entirely from what is observed on the network, and not from any internal state of the monitoring device (such as a local counter or clock within the monitoring device). According to this arrangement, two different monitors observing the same network traffic can compute the same signature without any coordination between them. The central collect can then perform any of a variety of actions upon the signature data. In the simplest

case, the central collector can compare the signatures on the received reports and discard any that are duplicates.

The units of a report, to which a particular signature pertains, are indivisible in the sense that the report will either be retained in its entirety, if unique, or discarded in its entirety, if a duplicate. The signature is computed in such a way that that its uniqueness is in one-to-one correspondence with the uniqueness of the network traffic being monitored. If two monitors generate reports based partly on the same network traffic, and partly on some traffic that is observed only at one or the other monitor, then those reports should be treated as duplicates, and one or the other is discarded. In such situations, it is not possible for the central collector to separate the contributions of the duplicate traffic from that of the unique traffic. Accordingly, the monitors must compute the same signature for those two reports.

The signature is also formed to be as small as possible to facilitate its efficient transmission and storage. In other words, the signature should consist of as few bytes of data as possible. This is because the central collector will have to keep a memory of signatures that it has already received, in order to determine if a newly-received report is a duplicate of a previous one. In order to minimize the size of this memory, or conversely to increase the time span covered by the stored items, the size of the individual signatures should be minimized. A minimum size of the signature is determined by the need to prevent collisions, i.e. cases where the same signature results from traffic that is actually unique, and not a duplicate. In the present application, collisions will lead to false detection of duplicates, and therefore the discarding of reports that should have been retained.

In general terms, a signature meeting the above requirements can be computed by a cryptographically strong hash function operating on selected parts of the network traffic that contribute to the report to which the signature is attached. A hash function takes input bits and produces output bits that vary according to the changing input bits. A cryptographically strong hash function has the property that changing any bit (or part) of the input data changes the resulting output result. Moreover, any random selection of input data values will, with very high probability, produce a uniform distribution of different result values. The hash function therefore allows for the reduction of a large number of bytes of data -- i.e. the portion of the observed traffic sufficient to establish the uniqueness of the observation -- to a relatively compact signature value.

Example hash functions include MD2, MD4 and MD5, details of which are further described in the following documents: B. Kaliski, "The MD2 Message-Digest Algorithm," Network Working Group, Request for Comments: 1319, RSA Laboratories, April 1992 (see Internet reference ftp://ftp.isi.edu/in-notes/rfc1319.txt); R. Rivest, "The MD5 Message-Digest Algorithm," Network Working Group, Request for Comments: 1321, MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992 (see Internet reference http://www.rfc-editor.org/rfc/rfc1321.txt); M.J.B. Robshaw, "On Recent Results for MD2, MD4, and MD5," RSA Laboratories Bulletin, Number 4, November 12, 1996; each of which are hereby incorporated by reference.

Note that the MD2 algorithm might be employed for high throughput applications. However, in terms of the following description (and for example purposes only), the MD5 algorithm is used. The MD5 hash function can generate a 128 bit (16 byte) signature from any amount of input data. The hash function generally processes the input data one byte at a time, with the result changing for each new input byte. The same sequence of input bytes will produce the same output result. For many applications, a smaller signature will suffice. A 64 bit (8 byte) signature, for instance, would result in about a one-in-ten-billion rate of false duplications when comparing each report against a billion stored signatures. If a lower false-alarm rate is needed, then more bits (i.e. up to the 128 bits) could be employed.

It is important to note that false alarms only result in discarding unique reports, and not in falsely retaining duplicate reports. The methods further detailed below are conservative in the sense that duplicates will always be discarded, and occasionally unique reports will also be discarded. When a good hash function is used to compute the signature, a smaller signature can be obtained from a larger one by simply truncating the result, and using a portion of the signature bits as appropriate.

The "signatory data," or signature, will consist of selected parts of the network traffic that will allow for identifying the uniqueness of the reports from the monitors. The reports of interest pertain to communication between a specific pair of computers, usually identified as a client and server. The network addresses of the two computers are therefore included in the signatory data. Network protocols typically include provisions for multiple virtual network connections, or ports, to each computer. The signatory data can therefore include the two port numbers identifying these network connections. Furthermore, the monitoring is

arranged so that each report (or set of reports) is triggered by the transfer of some specific data sequence (or one of a set of sequences) on the network. If the required sequence is observed, then the report will be generated. If the required sequence is not observed, then the report will not be generated.

Many network protocols additionally provide some kind of a serial number that labels each byte or packet of data that is transferred. For such protocols, the signatory data will include the serial (or sequence) number of a well-defined byte of the data sequence that triggered the report. Note that it would also be possible to include some of the actual data being communicated between the computers in the signatory data. However, this might not prove to be very useful because in many protocols it is possible, and even common, to request and send the same data multiple times, and those multiple transmissions should not be considered duplicates. The central collector cannot (practically) be configured to remember all signatures that have been reported to it, as the allocated memory would eventually fill up. However, data in computer networks cannot be "in flight" (i.e., in-transition across a network connection) for more than a reasonable period of time. If two monitors do report on the same traffic, network realities dictate that they will do so within this reasonable period of time, which is on the order of seconds or minutes (and not hours or days). Therefore, along with the signature, each report includes a time-stamp, and the central collector can be configured to discard signature information after a time period selected to be long enough to accommodate the network in-flight time, plus possibly delays in generating a report and forwarding it to the monitor. As the central collector receives report information -- and the signature data relating to those reports -- any of a variety of methods can be applied to analyze the data, and/or eliminate duplicate reports.

Referring now to Figure 1, a block diagram is shown of a representative network structure 100, which employs the monitors and central collector scheme of the present invention. Typically, any given network will have a hierarchical structure or arrangement of communicative devices. In this instance, routers, terminal nodes and users are shown, but many other devices might also be used. Routers are used to concentrate network traffic, and to send the traffic to another set of users. A first router 102 is shown connected to second and third routers 104 and 106. Each router 104 and 106 includes a series of terminal nodes, i.e. 108, 110, 112, 114, 116, and 118. The terminal nodes might end in user's computers 115

and 119, or network switches that have a number of computers of devices hanging off of them. This arrangement might be connected to yet another network (with a similar arrangement), hence the hierarchical arrangement.

In general, it is desired to capture and analyze data related to some part of the hierarchy, wherein the entire hierarchy might be representative of the Internet. A plurality of monitor devices (labeled "M") are shown as elements 120-130 interconnected along various parts of the network. The monitors are placed at strategic locations, and thus allow for detecting the data at various points in the hierarchy. Each monitor is typically a software device that can run on any machine. However, certain specifications or requirements might be put on PCs (or other devices) to run the software. Accordingly, the software and hardware device are often collectively referred to as a "box." The monitor is physically attached to the network connection of interest. In general, the traffic is routed fairly directly from one point to another (despite the ability of IP packets to flow anywhere). As a result, physical monitoring of point-to-point connections in the network can result in valuable analysis of traffic flows.

In certain instances, the monitors will need to be attached to network connections in multiple places, wherein monitors 124 and 126 show such instances. The monitors collect information pertaining to the traffic on the various network paths. Each monitor simply observes the packets going by, and does not do anything to hinder their progress. The monitors then communicate their observations with the central collector 140 through associated connections, wherein such connections might be hardwired, fiber, wireless, or the like.

Path element 160 shows an example data packet being transmitted between terminal 108 and terminal 110 through router 104. Note that monitor 120 and monitor 122 both observe the packet as it travels over the network connections. Prior to the present invention, there was no way to label or mark the observation of the data packet to show that it has already been observed. Note also that the pattern of travel for the packets is not regular. The example data packet 160 might have traveled over any path on the network to reach terminal 110, and could have been observed by any of the other monitors. If regular patterns exist, then the system could simply compensate for the regularity of observing any set of packets once, twice, or three times, etc. In a larger network, it becomes even more difficult to predict

with any regularity the observation of packets. It therefore becomes necessary to mark or label the packets in order to report their observance only once.

The prior art generally exists as any such network without the present solution for uniquely identifying data packets. Such networks will produce a duplicate report if a packet flows by two separate monitors. Prior solutions have included filtering, so that only particular monitors observe data with a certain range of IP addresses. For instance, if the system monitors some of the data at one monitor, and a non-overlapping set of data at another monitor, then no duplicate reports will be generated. Hence, the central collector can be made to decide that data transmitted from one point to another (or originating from some point) will be observed (and billed for). For traffic flowing in other directions, the system assumes that another monitor will detect and charge for such data. Problems exists with such a solution because it does not provide for adequate assurances for monitoring and billing of all data that might flow across the entire network. For example, data from a previously-known IP address may not be reported by any monitor. Any solution should eliminate duplicates automatically, and in a localized manner, without the collector having to know what the IP addresses are on one side of the network, versus IP addresses on the other side of the network. Complicating this filtering solution further is the networking practice of frequently changing the IP addresses as different connections are terminated, and others are set up.

The overall solution therefore involves putting an identification stamp, or signature, on the reports of the data packets, and thereafter employing specialized methods to eliminate the duplicate reports. Note that if a statistical analysis of a network were able to show that (approximately) 5% of the reports were duplicates, and the network reports could be compensated to correct for this percentage, then this situation might not warrant much concern. However, for billing purposes, duplicates cannot readily be tolerated. Even for such a small percentage of duplicates, a user will not generally tolerate being billed twice for any particular service. Additionally, a solution should provide for a conservative method in the sense that on rare occasions non-duplicate reports may be discarded, but a duplicate will never be retained. Hence, if certain monitoring data is discarded, but it is erroneously discarded because the data is not a duplicate, then a representative result might be that the user is not billed for that one service. Users are not likely to notice (or care about) such non-

billing discrepancies. Moreover, the service-providing company will not be financially hindered (due to the uncollected revenues for services, or the like) because the present solution provides for a very low chance of discarding non-duplicate reports. Any lost revenues are easily compensated, for instance, by not having to track down and deal with duplicate billings to a customer.

The present system uses the monitors to serve as a data reduction device. This is important because not all of the data can be sent back to the central collector due to the potentially high flow rate. It would also prove to be unwieldy to store all of the past data -- to compare it against incoming data -- in order to determine if duplicates exist (i.e. hundreds of gigabits of memory might be needed). Moreover, even if the system had the resources to look at all of the data packets, and be presented with the basic problem of analyzing whether a first and second data packet are the same, there might still be time delay problems associated with such monitoring. For instance, the monitors might be in the same room, or they might be in different parts of the country. Hence the time that one monitor sees a data packet might be very different from the time when another monitor sees that same data packet. If the time difference is great, the packets are not likely to be duplicates, despite their similar appearance.

To further reduce the level of data to be handled by the central collector, the present invention provides for tracking data sessions, as opposed to individual data packets. A user might be engaged in (for instance) a web browsing session, an email, or an FTP session. Each such session has a "start" and an "end" with several in-session reports being associated with each session. Given the present state of technology, a monitor might observe 1.5 million data packets per second across a data connection. Certainly, the monitoring boxes and the central collector could be configured to handle such loads, and it is within the spirit of the present invention to be able to track individual data packets. However, by using only sessions, a lesser amount of data (e.g., 100,000 sessions) would need to be tracked over that same time frame. This, of course, would prove to be more manageable.

When tracking sessions, it becomes necessary to distinguish one session from another session. Each session might consist of a number of data packets. One session monitored at one point on the network might have more total data packets than that same data session monitored at a different point. Even though the sessions have different data packet totals,

they are essentially the same session. For instance, certain sessions might have extra pages mixed into the content. Networks are also inherently unreliable, and if a packet does not make it through, then it is often retransmitted at different points along the network. With the different servers or devices behaving differently along the various parts of the network, data sessions monitored at two different points will often be different in length. Hence an overall data session might be a duplicate, even though it is not strictly identical at the byte level to a compared session.

Referring now to Figure 2, a representative block diagram 200 is shown to demonstrate how a session can appear to be different when monitored at a different points in the network. A first computer (or server) 202 (i.e. client device like a browser, or the like) is shown communicating with a second computer (or server) 204 (i.e. a content server device) over a network 206. In a client-server relationship, device 202 might be a client computer operating a browser. The server 204 might be a content portal. The network connection passes through several routers, switches, etc., illustrated for example by Router 1 (220), Router 2 (222), and Router 3 (224). There are multiple routes that a packet might take between client 202 and Server 204. As a particular (non-limiting) example, Server 202 is shown sending 4 data packets to the server 204. Server 204 might then send 8 data packets back to server 202. A first monitor 208 is shown oriented to observe packets flowing to and from client 202. A second monitor 210 is shown oriented to observe packets flowing to and from server 204, but only observes those packets traversing one of the network connections to Router 1 (220). The monitors 208 and 210 communicate with the central collector 212. In this example, 6 of the 8 packets from Server 204 are observed by monitor 210; the other 2 packets take a different route. All 8 packets are observed by monitor 208. Accordingly, while the total packets observed at each monitoring point are different, the overall session is to be considered the same.

In a client-server relationship, a session is generally defined by certain synchronization (SYNC) and acknowledgement (ACK) packets being sent between the devices. The present invention takes advantage of certain information known to be contained in these SYNC and ACK packets in order to create a unique identifier for the data session (or alternatively the data packets, or set of data packets). Referring now to Figure 3, a representative block diagram 300 is shown of certain packets for establishing a session. A client device 302 is

shown communicating over a data connection 306 with a server device 304. A monitor 308 is associated with the connection 306 to observe data packets flowing across the connection. For two normally networked computers, each computer has an IP address which is used for communicating with that computer. The client IP address and server IP address are unique to the session, and therefore used in the identifier.

Port numbers are also established and agreed upon by the client and server. A port number is a unique network link, or a software abstraction that makes each link unique for that session. For instance, several programs might be running on a computer. Each packet comes in with an IP address, so that it arrives at the correct computer. The packet needs additional direction, however, to be sent to the proper application running on that computer. This is achieved via the assigned port number for that packet. Note that servers tend to have fixed port numbers, i.e. port 80 for web server traffic, port 25 for email traffic, and so forth. Each client computer will dynamically assign a port number, and this is usually done by arbitrarily picking an available port. Hence, in comparing any two packets, if the client port and server port are the same, then the packets might be from the same session. Two different sessions, however, might still have been assigned similar IP addresses, a client port number (arbitrarily assigned), and a server port number (fixed), and therefor more information is ideally needed to make a unique identification.

For TCP communications, each packet has an associated sequence number. This sequence number is a 4 byte (or 32 bit) value that indicates how many bytes have been sent since the connection was established, starting with some pseudo-randomly chosen number. Referring again to Figure 3, a client first sends a SYN packet 310 to the server 304. This SYN packet 310 includes the client IP address, the server IP address, the client port number, the server port number, and the client sequence number. The server 304 then sends back a server SYN/ACK packet 312, which includes the client IP address, server IP address, the client port number, the server port number, the server sequence number, and an acknowledgement of the client sequence number. With this information, the client knows that the sequence number has been acknowledged. Thereafter, the client will send back an client ACK packet 314, which includes the client IP address, the server IP address, the client port number, the server port number, the client sequence number, and an acknowledgement of the server sequence number. Once these TCP packets have been exchanged, a session is

underway and continuing data relating to the session (i.e. email, webpages, etc.) will be sent as part of this session. This will continue until the session is terminated.

Referring now to Figure 4, a block diagram 400 is shown of representative information that can be used to form a unique signature relating to a data packet. It is important to uniquely identify each of these packets as they go by. While counters exist (i.e. sequence numbers), the system cannot simply count the packets as they might transition at different times, due to network delays and the like. Therefore it is important to collapse the information down into a unique "nugget" or "digest" of information, herein referred to as a signature.

A hash function 414 is used to convert the input bytes into this unique signature 416. The hash function might include MD5 (Message Digest Algorithm #5), or the like, details of which are further described in the documents referenced above. This hash function can generate a 128 bit (16 byte) signature from any amount of input data, with the result being cryptographically strong. That is, given a particular input (sequence of bytes), it is very unlikely that another randomly-chosen input will produce the same signature. Also, there is no simple relationship amongst sets of inputs that result in the same signature. The result is that the signature is a good digest of the bytes that were fed into the algorithm, with few collisions: only very rarely will two signatures be equal unless they were generated from identical inputs. The hash function generally processes the input data one byte at a time, with the result changing for each new input byte. Conversely, the same sequence of input bytes will produce the same output result. When identical signatures are produced for different inputs, a collision is said to occur.

In the present example, the hash function 414 receives as input bytes the client IP address 402, the server IP address 404, the client port number 406, the server port number 408, and the client sequence number 410. An optional input might include the server sequence number 412. This optional input would provide for an even more unique set of input bytes. The input bytes are organized or assembled into an input string to the hash function. The hash function 414 produces a resulting output string, or signature 416. For the aforementioned MD5 hash function the signature result will include up to 16 bytes.

Thereafter, the result can be truncated as shown in step 418. By truncating the result, the signature is shortened and the amount of data being sent to the central collector is decreased. Truncating the result will not have a detrimental effect on the overall ability of the present invention to detect and/or eliminate duplicates at the central collector. For instance, even by truncating the signature in half, i.e. down to 8 bytes, the collision rate will still be very rare. (i.e., 8 bytes = 64 bits, with the collision rate being approximately one in 2^64; this roughly equals a billion-billion sessions). Moreover, with the conservative method of the present invention, even if a collision occurs, the data report for that session is simply discarded. Hence, a "valid" data report is lost as a (false) duplicate once out of every billion-billion record comparisons. For a billing system, this means that a company might lose revenue on a very small percentage of billables. Such loses are much more preferable to double-billing a customer, as it is important to preserve customer trust and loyalty. The signature can be truncated to any degree needed, with the disadvantage being a higher collision rate, and the advantage being that less data is attached to each report as an identifier.

The present system thereby calculates the signature by running the MD5 (or other function) over 16 or 20 bytes of data -- i.e. 8 bytes of IP addresses (4 each from client and server), 4 bytes of port numbers (2 each from client and server), 4 bytes of client sequence number, and optionally 4 bytes of server sequence number. Figure 4A shows a block diagram that demonstrates the formation of an enhanced signature. The 128 bit (16 byte) hash function output 450 is returned by MD5 (or other hash function). This output value is truncated by selecting a portion of the bits 452. While any number might be selected, this example uses 56 bits. Next an application-protocol-specific 8 bit tag 454 is added, thereby producing a 64-bit signature 458.

The refinement of adding the 8 bit protocol tag to the signature ensures that any collisions (or false duplicates) that do occur will be between reports concerning the same application-level protocol. This refinement is motivated by the observations that certain reports are more valuable than others, that the inadvertent loss of certain reports might cause greater problems, or incur greater costs, than others, and that the more-valuable reports often relate to less-common protocols. For instance, login data might be more important for billing purposes than data within the heart of a user's on-line session. Such valuable pieces of data generally make up a smaller fraction of the total traffic. By using a protocol tag, the system

insures that the more valuable reports will be inadvertantly discarded only for collisions with others of the same protocol. Such in-protocol collisions are even more rare than out-of-protocol collisions, and therefore the more valuable data reports are rarely (if ever) discarded. (Note that the collision frequency is roughly proportional to the square of the occurrence frequency.)

Referring again to Figure 4, step 420 shows the resulting signature being attached to the data report (or set of data reports). This signature will be attached to each report sent out from the monitor. Step 422 shows the reports thereafter being sent to the central collector, along with a timestamp indicating the time of the occurrence of the network event being reported. With each report now uniquely identified, step 424 shows a duplicate elimination process being applied to the reports that are sent to the central collector.

It should further be noted any set of parameters might be used to generate a signature via a hash function. However, sometimes data packets are dropped, or lost, while transitioning across the network. By using the input data described above, certain information has more than one chance of being observed. First instance, the first packet 310 includes all five pieces of information needed for the basic signature calculation (i.e., client and server IP addresses, client and server port numbers, and client sequence number). The second and third packets 312 and 314 includes all six pieces of information needed for the optional signature calculation (adding the server sequence number). Hence, a monitor has more than one opportunity to observe the key identification information. This is a valuable feature in case a packet is missed (or dropped) by a monitor.

Many different sessions can be going on at the same time across any network connection. The monitor looks for a start and end of each session. Once a session is established, an initial report is sent out to the central collector. Thereafter interim update reports are sent while the session is still occurring. A final report is sent when the session ends. If a monitor does not see the initial packet exchanges necessary to establish a session, then the monitor will ignore data associated with that session.

In summary, the method described above has been implemented for network traffic using the TCP protocol. This is meant for example purposes only, to demonstrate the applicability of the present methods to other protocols that might contain a set of unique

identifier information. Being based on IP, TCP addresses the interacting computers via IP addresses, which are unique 4 byte numbers. These are used in the signatory data. TCP also uses port numbers (as discussed) above, which are also included in the signatory data. TCP is also a so-called connection-based protocol, in that the data transactions are structured in such a way that there is an on-going logical connection between the client and server computers, even though the data on the network is actually transferred in small discrete packets. The implementation is based on doing duplicate elimination over the entire connection. That is, for a particular connection, either all the reports from a monitor will be accepted as unique, or all will be rejected as being duplicates because another monitor has reported on the same connection (i.e. has sent reports with the same signature). The TCP protocol further includes a sequence number, which is a serial number that starts from a pseudo-random value when a connection is set up and thereafter counts the bytes that have been sent over the connection. There is a special "handshake" transaction that occurs when the connection is set up. The sequence number of this transaction -- specifically, the TCP sequence number corresponding to the first byte of the packet sent by the client with the "SYN" flag set -- is included in the signatory data. The final component of the signatory data is an integer that specifies the association with an application-level protocol. For each report or set of reports, the signatory data is assembled in a consistent ordering. The hash function is applied to the data and the result is truncated to the desired number of bits of signature. That signature is attached to the report, or to each report in a set, and the report(s) are transmitted to the central collector along with a timestamp.

The data identification procedure can also be applied to other protocols as well. With UDP (for instance), no logical connection is set up between the communication devices. There is no handshaking routine, and no sequence numbers. With this protocol, the client will send data to the server, but have no way of knowing if that data arrived properly. The server will send information back, and also not know if the data was received. Despite the unreliable nature of this protocol, a signature can still be calculated using the IP numbers and the port numbers of the client and server devices. This signature can then be applied to data reports and used for analysis and/or duplicate elimination purposes. While the lack of sequence numbers provides a set of data that is less unique, and might involve more collisions, the addition a protocol-specific flag as described above will further assist in providing a unique signature.

Similarly, signatures could be computed for TCP sessions without including the sequence number. This would increase the frequency of false duplicates, but would allow a monitor to begin tracking a session in the middle, without observing the initial client/server SYN transaction. Conversely, UDP and/or TCP signatures could be computed from individual data packets using the 16 bit serial number in the IP header. This would increase the computation required, but allow for duplicate elimination on a more fine-grained basis.

Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. For instance, the described generation of a unique signature in relation to TCP or UDP is meant to be applicable to other networking protocols, as well as to reports generated indirectly such as by parsing log files or other records. By taking unique elements of the transmitted data and applying a hash function on the compiled data elements, a unique signature can be generated for the data. Therefore, the described embodiments should be taken as illustrative and not restrictive, and the invention should not be limited to the details given herein but should be defined by the following claims and their full scope of equivalents.